REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
May 13, 1994

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
A Fast Numerical Method for Isothermal Resin Transfer Mold Filling

5. FUNDING NUMBERS
DAAH04-93-G-0207

6. AUTHOR(S)
R.S. Maier, T.F. Rohaly, S.G. Advani, K.D. Fickie

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Army High Performance Computing Research Center
University of Minnesota
1100 Washington Avenue South
Minneapolis, MN 55415

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
ARO 32102.2-MA

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

An efficient numerical scheme is presented for simulating isothermal flow in resin transfer molding (RTM). The problem involves transient, free-surface flow of an incompressible fluid into a nondeforming porous medium. A new variant of the control volume finite element (CVFE) algorithm is explained in detail. It is shown how the pressure solutions at each time step can be obtained by adding a single row and column to the Cholesky factorization of the stiffness matrix derived from a finite-element formulation for the pressure field. This approach reduces the computation of a new pressure solution at each time step to essentially just two sparse matrix back-substitutions. The resulting performance improvement facilitates interactive simulation and the solution of inverse problems which require many simulations of the filling problem. The computational complexity of the calculation is bounded by $O(n^{2.5})$, where n is the number of nodes in the finite element mesh. A 100-fold speedup over a conventional CVFE implementation was obtained for a 2213-node problem.

14. SUBJECT TERMS
RTM, Resin Transfer Molding, Free Surface Flow, Cholesky Factorization, Control Volume Finite Element Method

15. NUMBER OF PAGES
21

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

A Fast Numerical Method for Isothermal Resin Transfer Mold Filling

Technical Report

R. S. Maier, T. F. Rohaly
S. G. Advani, and K. D. Fickie

May 13, 1994

U.S. Army Research Office
Contract DAAH04-93-G-0207

Army High Performance Computing Research Center
University of Minnesota
1100 Washington Avenue South, Minneapolis, MN 55415

Approved for Public Release; distribution unlimited.

A Fast Numerical Method for Isothermal Resin Transfer Mold Filling

R. S. Maier [*] T. F. Rohaly [†] S. G. Advani[‡]
K. D. Fickie [§]

May 13, 1994

Abstract

An efficient numerical scheme is presented for simulating the isothermal flow in resin transfer molding (RTM). The problem involves transient, free surface flow of an incompressible fluid into a nondeforming porous medium. A new variant of the control volume finite element (CVFE) algorithm is explained in detail. It is shown how the pressure solutions at each time step can be obtained by adding a single row and column to the Cholesky factorization of the stiffness matrix derived from a finite-element formulation for the pressure field. This approach reduces the computation of a new pressure solution at each time step to essentially just two sparse matrix back-substitutions. The resulting performance improvement facilitates interactive simulation and the solution of inverse problems which require many simulations of the filling problem. The computational complexity of the calculation is bounded by $O(n^{2.5})$, where n is the number of nodes in the finite element mesh. A 100-fold speedup over a conventional CVFE implementation was obtained for a 2213-node problem.

[*] Army High Performance Computing Research Center, 1100 Washington Avenue So., Minneapolis, MN 55415, maier@arc.umn.edu.

[†] Army Research Laboratory, Aberdeen Proving Grounds, MD.

[‡] Department of Mechanical Engineering, University of Delaware, Newark, DE.

[§] Army Research Laboratory, Aberdeen Proving Grounds, MD. 21005; fickie@arl.mil

1 Resin-Transfer Mold Filling Problem

Resin transfer molding (RTM) is an emerging manufacturing technology well-suited for fabricating large structural components made of composite materials. Since the process involves matched metal tooling, the technique seems ideal for situations requiring close tolerances. Construction of aircraft structures and vehicle components fit this characterization. Furthermore, liquid injection molding represents one of the most economical means of manufacturing. RTM is an adaptation on a process widely used for plastics. Instead of injecting into an empty cavity, the mold is packed with a woven fiber preform. The RTM process has two main stages: filling the mold with a resin/catalyst mixture and curing the part.

At present, most of the difficulties of incorporating RTM revolving around the filling. To create an acceptable composite part requires the preform to be completely impregnated with resin. This largely controlled by the fluid dynamics of the resin flow into the fiber reinforcement. The conditions which most strongly influence the the flow are: mold geometry, resin rheology, preform permeability, and location of the injectors and vents. The first three are typically determined by the part design itself; the last one is a manufacturing consideration. Incorrect placement of the injectors and vents for a given geometry and resin/preform system will create dry spots in the cured part.

To enhance the economic viability of RTM applications, it is desirable to evaluate mold design prior to mold construction, via computer-based methods. Predictive modeling of resin flow through a fiber preform is currently an important priority in mold design and evaluation, because of the need to predict fill times and wet-out patterns. Ongoing research also includes control algorithms for the filling stage, using networks of embedded sensors and a fast filling simulation [1, 2].

Simulation of the RTM process may be *isothermal* or nonisothermal, depending upon whether temperature effects are accounted for in the model. During filling, resin viscosity is affected by temperature variations. During curing, gel times are affected by temperature profiles. For a mathematical formulation of the nonisothermal RTM process, see [4]. We focus on the isothermal case under the assumption of minimal temperature variation during filling.

The isothermal RTM filling problem is a transient, free-boundary problem of predicting the position of the resin flow front in the porous medium as a function of injection pressures and time. The resin is assumed to be nearly compressible and to display Newtonian behavior. The fiber preform is assumed to be non-deforming. It is assumed that Darcy's law governs the relation between resin velocity, $\mathbf{v}$, and pressure p, such that

$$\mathbf{v} = -\mu^{-1} K \nabla p, \tag{1}$$

where μ is the viscosity and K is a tensor representing the permeability of the fiber preform. Preforms are usually constructed from several layers of fiber mat oriented in different directions. Permeabilities are experimentally calculated for mat samples and reported in terms of the principal directions of the mat. Thus, the average (through-thickness) permeability is a function of several factors, see for example [8]. Since permeabilities along the principal axes can easily differ by an order of magnitude, the ability to specify K on a local basis is essential in simulation.

Let: $\mathbf{\Omega} \subset \mathbf{R}^3$ define the interior of the mold; Γ_w the impermeable mold walls; Γ_h the constant displacement injectors; and Γ_g the constant pressure injectors; so that the complete assembly includes

$$\overline{\mathbf{\Omega}} = \mathbf{\Omega} \cup \Gamma_w \cup \Gamma_h \cup \Gamma_g. \tag{2}$$

Let: $\Omega(t) \subseteq \mathbf{\Omega}$ denote the filled portion of the mold interior at time t and $\Gamma_s(t)$ the free surface at time t. On the air side of the surface, the capillary fringe is neglected and a constant pressure (e.g. atmospheric) is assumed in the unfilled portion of the mold $\{\mathbf{\Omega} \setminus \Omega(t)\}$.

The isothermal RTM filling problem is to find for any $t > 0$, $\Gamma_s(t) : \overline{\mathbf{\Omega}} \mapsto \mathbf{R}^3$ and $p : \Omega(t) \mapsto \mathbf{R}$ such that

$$\begin{aligned}
\nabla \cdot (\mu^{-1} K \nabla p) &= 0 && \text{in } \Omega(t) && (3)\\
\mathbf{n} \cdot (\mu^{-1} K \nabla p) &= 0 && \text{on } \Gamma_w && (4)\\
\mathbf{n} \cdot (\mu^{-1} K \nabla p) &= h(\mathbf{x}) && \text{on } \Gamma_h && (5)\\
p &= g(\mathbf{x}) && \text{on } \Gamma_g && (6)\\
p &= 0 && \text{on } \Gamma_s(t) && (7)\\
\mathbf{n} \cdot (\mu^{-1} K \nabla p) &= -\mathbf{n} \cdot \frac{d\Gamma_s}{dt} && && (8)
\end{aligned}$$

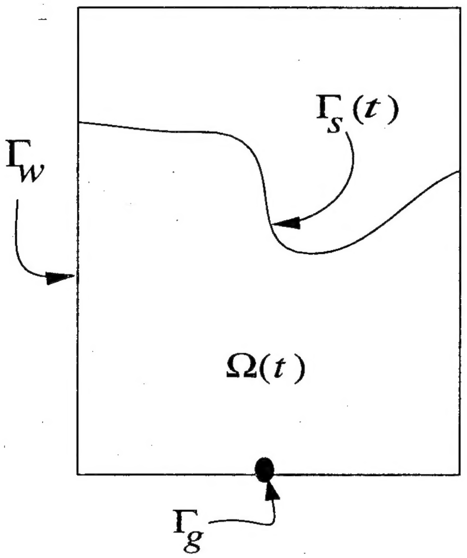

Figure 1: **Two-Dimensional RTM Mold. Resin flow front is a free surface.**

where $\mathbf{n}$ denotes the vector normal to Γ. The quantity $f = \mathbf{n} \cdot \mathbf{v}$ is defined as the flux normal to the boundary. The mold- filling problem is analogous to the *Stefan problem*, a class of free boundary problems used in modeling the melting of solids and crystallization of liquids [3, 15].

2 Filling Algorithm

The mold filling problem in equations (3) - (8) may be solved with a variety of numerical schemes, including fully implicit methods which solve simultaneously for free surface location and pressure, as well as more conventional semi-implicit methods which solve for pressure implicitly and satisfy the free surface condition (8) by an explicit method. We follow the latter approach, suggested by many researchers [5, 9, 4, 6], using a finite element solution to obtain pressures at discrete values of t followed by an explicit control volume scheme for updating the position of the free surface. The control volume scheme involves domain discretization into discrete subvolumes, where each such control volume contains one node of the finite element mesh. Fluid flux across control volume boundaries is calculated from the pressure solution. Net inflow to a control volume is tracked as the "fill fraction". When the volume is filled, the node contained in the volume is considered part of the next pressure solution, as summarized below.

1. *Find FEM pressure solution.* At the beginning of a new time step, the pressure field is calculated over the filled control volumes.

2. *Calculate volume flux.* Darcy velocity and flux at the control volume boundaries are computed from the pressures calculated in step 1.

3. *Locate the free surface.* The time step is calculated as the minimum Δt required to fill a control volume using the flux calculated in step 2. The filled volume becomes part of the pressure field, moving the free surface to a new position.

2.1 Finite Element Pressure Solution

At each time step, a free surface location $\Gamma_s(t)$ is given from the previous step. The problem of solving equation (3) subject to (4)-(7) is analogous to the classical heat conduction problem, which has the following weak formulation. Define the trial function and weight function spaces,

$$\begin{aligned} U &= \{p \mid p \in H^1,\ p|_{\Gamma_s(t)} = 0,\ p|_{\Gamma_g} = g(\mathbf{x})\} && (9) \\ V &= \{w \mid w \in H^1,\ w|_{\Gamma_g \cup \Gamma_s} = 0\}. && (10) \end{aligned}$$

Given g, h, find $p \in U$ such that for all $w \in V$,

$$\int_\Omega (\nabla w)^T(\mu^{-1}K\nabla p)d\Omega = \int_{\Gamma_h} whd\Gamma_h. \qquad (11)$$

The Galerkin formulation (in two spatial dimensions) is based on a triangulation of the mold into nel elements,

$$\overline{\mathbf{\Omega}} \approx \mathbf{T} = \cup_i T_i, i = 1, ..., nel,$$

where the T_i are closed triangles with nodes (vertices) $\mathbf{x}_j = (x_j, y_j), j = 1, ..., n$ (see Figure 2).

We have chosen to work in the space of piecewise linear functions. Let $\tilde{p}$ be the finite dimensional approximation to p, such that $\tilde{p} \in U$ is a polynomial of degree one over each triangle. The vector $\mathbf{p}$ will denote the nodal values of $\tilde{p}$, i.e.,

$$[\mathbf{p}_1, ..., \mathbf{p}_n]^T = [\tilde{p}(\mathbf{x_i}), ..., \tilde{p}(\mathbf{x_n})]^T$$

and $\tilde{p}(\mathbf{x}) = \sum_{i=1}^n N_i(\mathbf{x})\mathbf{p_i})$, where the shape functions $N_i(\mathbf{x}_j) = \delta_{ij}, i, j = 1, ..., n$ are piecewise linear. The Galerkin formulation is then

$$\sum_{j=1}^n \left(\int_\Omega \nabla N_i \cdot (\mu^{-1}K\nabla N_j)d\Omega\right) \mathbf{p}_j = \int_{\Gamma_h} N_i hd\Gamma_h, \quad i = 1, ..., n. \qquad (12)$$

The elements of the $n \times n$ pressure stiffness matrix A are defined

$$a_{ij} = \int_\Omega \nabla N_i \cdot (\mu^{-1}K\nabla N_j)d\Omega,\ i, j = 1, ..., n, \qquad (13)$$

and the forcing vector $\mathbf{b}$ as

$$\mathbf{b}_i = \int_{\Gamma_h} N_i hd\Gamma_h, \quad i = 1, ..., n.$$

In matrix notation,

$$A\mathbf{p} = \mathbf{b}, \tag{14}$$

and the approximate pressure solution is obtained by solving a linear system of n equations in n unknowns.

2.2 Control Volume Flux Calculation

The flux calculation approximates fluid velocities by the piecewise linear polynomials described in the previous section,

$$\mathbf{v}(\mathbf{x}) = \sum_{i=1}^{n} \mu^{-1} K \nabla N_i(\mathbf{x})\mathbf{p}_i.$$

We use a node-centered control volume discretization to calculate flux and fluid volume fractions. The control volume discretization is built on the triangulation $\mathbf{T}$ (see Figure 2) and forms a set of closed subvolumes C_i,

$$\overline{\boldsymbol{\Omega}} \approx \mathbf{C} = \cup_j C_i, i = 1, ..., n,$$

such that subvolume C_i contains vertex $\mathbf{x}_i$ and no other node. Denoting B_i as the boundary of C_i, the flow rate into C_i at time t is

$$q_i(t) = \int_{B_i} \mathbf{n} \cdot \mathbf{v}(t) dB_i. \tag{15}$$

where $\mathbf{n}$ is the unit normal to B_i.

We define B_i as the set of line segments connecting element centroids with edge midpoints, as follows (see Figure 3). Associated with B_i is the set of elements $E_i = \{T_j \mid \mathbf{x}_i \in T_j\}$. Without loss of generality, denote the vertices of T_j as $\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}$. Define the centroid of T_j as $\mathbf{c_j} = \frac{1}{3}\sum_{k=i}^{i+2} \mathbf{x}_k$ and the edge midpoints, $\mathbf{m}_{j1} = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_{i+1})$ and $\mathbf{m}_{j2} = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_{i+2})$. The segments of B_i in T_j are denoted by $\overline{(\mathbf{c}_j, \mathbf{m}_{j1})}, \overline{(\mathbf{c}_j, \mathbf{m}_{j2})}$. Then q_i can be written as an algebraic sum,

$$q_i = \sum_{T_j \in E_i} (l_{j1}\mathbf{n}_{j1} + l_{j2}\mathbf{n}_{j2}) \cdot \sum_{k=1}^{3} \mu^{-1} K \nabla N_k \mathbf{p}_k \tag{16}$$

where $\mathbf{n}_{j1}$ is the unit normal vector in the plane of T_j orthogonal to $\overline{(\mathbf{c}_j, \mathbf{m}_{j1})}$, and $l_{j1} = \|\mathbf{c}_j - \mathbf{m}_{j1}\|_2$ is the segment length. The coefficients of $\mathbf{p}$ in equation (16) are constant for all t and are assembled prior to the filling algorithm.

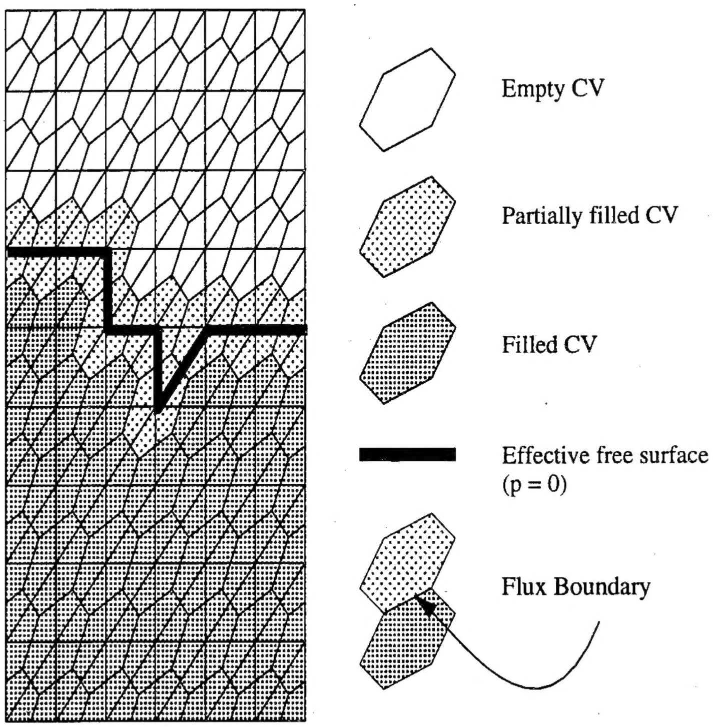

Figure 2: Triangulation and Control Volume Discretization of $\mathbf{\Omega}$.

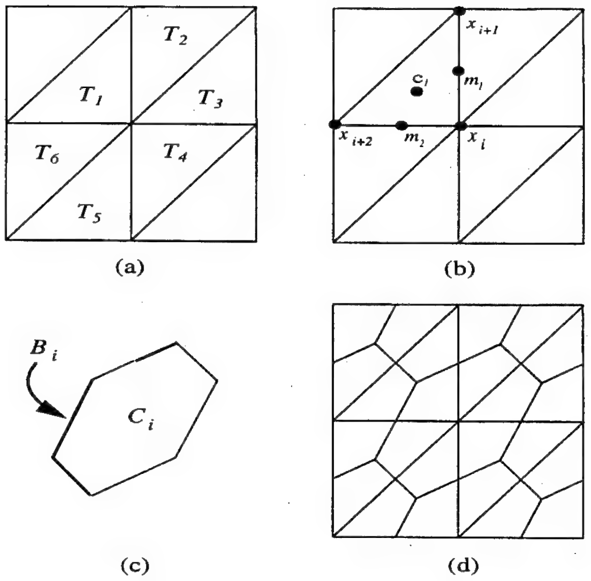

Figure 3: **Control Volume Discretization.** (a) The set E_i of elements supported by node i, (b) centroid and edge midpoints for T_1, (c) control volume C_i and bouundary B_i, (d) superimposed FEM and control volume discretizations.

2.3 Free Surface Location

A node $\mathbf{x}_i$ is included in the filled domain $\Omega(t)$ if control volume C_i has a fill fraction of unity. Let $S_i(t)$ denote the fill fraction of C_i at time t,

$$S_i(t) = |C_i|^{-1} \int_0^t q_i(t)\partial t, \tag{17}$$

where $|C_i|$ denotes volume adjusted for the porosity of the preform. At each time step, the fill fraction is updated explicitly. If $q_i(t)$ is the flow rate into C_i at time t, then

$$S_i(t + \Delta t) = S_i(t) + \Delta t q_i(t). \tag{18}$$

According to equation (7), the free surface location satisfies the relation $p|_{\Gamma_s} = 0$. We define the free surface location by the nodes in unfilled or partially filled volumes adjacent to filled nodes, i.e., $\mathbf{x}_j$ is on the free surface if (a) $S_j < 1$, and (b) $\mathbf{x}_j$ is adjacent to a node $\mathbf{x}_i$ such that $S_i = 1$. Observe in Figure 2 the free surface intersects partially filled control volumes and that control volume flux is not actually calculated at the free surface. To address this fundamental discrepancy, some researchers have developed local refinement schemes in the flow front vicinity, e.g. [10]. Such schemes improve the local accuracy of the flow front approximation. However, we believe that if high accuracy in flow front calculations is needed, then an alternative filling algorithm satisfying more rigorous mathematical convergence criteria should be considered rather than local mesh refinement.

2.4 Implementation

2.4.1 Properties of the Pressure Stiffness Matrix

The pressure stiffness matrix A in equation (14) has several important properties which lead to an efficient implementation of the filling algorithm. The first property is sparsity, due to the structure of the finite element mesh. The order of A is equal to the number of nodes, n in the mesh. The number of nonzero entries in A is equal to the number of edges connecting adjacent nodes. Since the nodes of a finite element mesh typically are connected to only a few other nodes, the number of nonzeroes is far less than n^2, usually a small multiple of n. The second and third properties of A are *symmetry*, $A = A^T$ and *positive-definiteness*, $\lambda(A) > 0$.

To demonstrate positive-definiteness, it is necessary to show that K is always positive-definite. Experimental permeability measurements are reported for principal mat directions as a diagonal matrix with strictly positive entries, $D = diag(k_{11}, k_{22}, k_{33})$. The tensor K is obtained by transforming from the principal directions of the mat to the Cartesian frame (or to local element coordinates), i.e., $K = CDC^T$, where C is a rank-3, orthogonal rotation matrix which projects the principal axes of the mat into the Cartesian or local coordinate system. Since pre or post multiplication by an orthogonal matrix preserves the spectrum of an operator, $\lambda(K) = \lambda(D) > 0$. Given $\lambda(K) > 0$, symmetry and positive definiteness of A is a standard result in the finite element literature; see, for example, [14].

Since A is symmetric and positive definite, it has a Cholesky factorization, $LL^T = A$, where L is a lower triangular matrix. It is also true that every submatrix of A inherits these two properties. Thus, if A is partitioned as

$$A = \begin{pmatrix} M & u \\ u^T & s \end{pmatrix} \tag{19}$$

then the matrix M has a Cholesky factorization $M = L_M L_M^T$ and

$$L = \begin{pmatrix} L_M & 0 \\ w^T & t \end{pmatrix} \tag{20}$$

where $L_M w = u$ and $t = (s - w^T w)^{\frac{1}{2}}$. As a result, the Cholesky factor of the stiffness matrix can be computed row by row. This is exactly the property required for an efficient isothermal filling algorithm.

2.4.2 Updating the Pressure Solution

Each time step Δt is calculated to fill one control volume. Filled volumes are considered part of the fluid phase and so the corresponding node becomes part of the fluid pressure calculation. The addition of a node to the pressure calculation corresponds to adding a single row and column to the stiffness matrix. The Cholesky factorization of the updated stiffness matrix can be updated directly, as in equation (20). The advantage is that the stiffness matrix A need only be factored one time, rather than reassembling and factorizing at every time step.

The stiffness matrix A is assembled and stored prior to filling. The full matrix is stored in an *adjacency structure.* The adjacency structure consists of n adjacency lists and corresponding nonzero coefficients. The ith adjacency list includes the indices of nodes which are adjacent to (share an edge with) node $\mathbf{x}_i$.

During the filling algorithm, nodes are added to the pressure field as control volumes are filled. Rows of A corresponding to these nodes are added to the Cholesky factor L using Equation (20). Note that these rows must be permuted to reflect the node ordering imposed by the filling sequence. The data structure used to store L is an *envelope* structure. For each row of the matrix, all entries from the first nonzero up to the diagonal are stored.

The envelope storage scheme is a standard data structure for sparse matrix factorization. This choice permits the use of existing numerical software [12] for updating the Cholesky factorization and computing intermediate pressure solutions at each time step, with only minor modifications.

In practice, the algorithm will often fill more than one node in a single time step, despite the fact that the time step is calculated to fill only one volume. This occurs most typically in regular discretizations because a tolerance is used to define the fill fraction constituting a "filled" control volume (e.g., 99%).

2.4.3 CVFE Algorithm

The CVFE algorithm requires an extensive set of inputs and initialization steps. For the case of a two-dimensional thin shell geometry in three dimensional space, these initialization steps include:

- specification of a triangulation $\mathbf{T}$,
- specification of a control volume discretization $\mathbf{C}$,
- specification of local permeabilities and element thickness,
- rotation of permeabilities to local element coordinates,
- calculation of adjacency data structure for A,
- assembly of A in adjacency structure,
- assembly of flow rate coefficient matrix.

The iterative part of the algorithm computes the filling sequence, using the integer arrays *perm* and *invp* to denote the ordering of A (the filling sequence) used in the Cholesky factorization. The notation $perm(i) = k$ means the original node k is the ith node in the new ordering. The element $invp(k)$ gives the position in *perm* where the node originally numbered k resides, i.e. $perm(invp(k)) = k$. We use the array subscript notation $b_i = b(i)$ interchangeably.

```
$t = 0$;  $l = 0$;  $m = 0$; filled = false;
while (not filled)

    compute forcing vector $\mathbf{b}$

    {add row(s) to pressure stiffness factor $L$}

    for $i = 1, ..., n$                          {add filled control volumes to $\Omega(t)$}
      if $S_i(t) = 1$ and $\mathbf{x}_i \ni \Omega(t)$,
        $m = m + 1$;                             {increment number of filled volumes}
        $perm(i) = m$;
        $invp(m) = i$;
        {scatter row $i$ from adjacency structure of $A$ to full vector}
        $\mathbf{w}(j) \leftarrow a_{ij}$,  $j = 1, ..., n$;
        {gather permuted row $i$ into envelope structure for $L$}
        $L_{ij} \leftarrow \mathbf{w}(invp(j))$,  $j = 1, ..., n$;
      end if
    end for

    {solve updated pressure system}

    $\mathbf{b}(perm(i)) \leftarrow \mathbf{b}(i)$,  $i = 1, ..., n$;       {permute forcing vector}
    update $L_{ij}$,  $i = l + 1, ..., m$ ,  $j = 1, ..., m$    {eqn (20)}
    solve $\mathbf{y} \leftarrow L\mathbf{y} = \mathbf{b}$;
    solve $\mathbf{p} \leftarrow L^T\mathbf{p} = \mathbf{y}$;
    $l = m$;                                     {updated dimension of $L$}
    $\mathbf{p}(invp(i)) \leftarrow \mathbf{p}_i$,  $i = 1, ..., m$;       {restore solution to original order}

    {update fill fractions}

    compute $q_i(t)$,  $i = 1, ..., n$;
    if $q_i(t) == 0,\ i = 1, ..., n$, STOP;        {mold cannot be filled}
    $\Delta t = \min_i[(1 - S_i(t))\ |C_i|\ /\ q_i(t)],\ i = 1, ..., n$;
    $S_i(t + \Delta t) = S_i(t) + \Delta t q_i(t),\ i = 1, ..., n$;
    $t = t + \Delta t$;
    if $S_i(t) = 1,\ i = 1, ..., n$, filled = true;

end while
```

The order in which control volumes are filled determines the *ordering* of A during the Cholesky factorization. A more efficient ordering could be obtained by using a symbolic factorization algorithm, such as reverse Cuthill-McKee [11] to find an ordering which reduces the maximum envelope bandwidth. Such a procedure reduces the number of nonzero elements in L and hence the computational effort in solving $Ly = b$ and $L^T x = y$. This possibility will be addressed in the context of further research on the application of direct factorization methods for large-scale RTM simulation and parallel computation.

2.5 Computational Complexity

This section analyzes the computational requirements of the filling algorithm as a function of the problem size. The analysis is based on several assumptions about typical models and is not a worst-case analysis of complexity.

The CVFE algorithm requires $O(n)$ iterations or time steps, one per control volume (in practice, more than one volume may fill per time step). Each iteration requires the four procedures as summarized below.

```
for $k = 1, ..., n$
    compute forcing vector $\mathbf{b}$              $O(n)$ operations
    add row(s) to $L$                        $O(n)$ operations
    solve updated pressure system            $O(k^{1.5})$ operations
    update fill fractions                    $O(n)$ operations
end while
```

Three procedures require $O(n)$ operations per iteration, or $O(n^2)$ operations for all iterations. One procedure, solving the updated pressure system, requires $O(k^{1.5})$ operations per iteration, where k is the iteration number. This procedure dominates the total computation and is explained below.

The updated pressure system requires several sparse-matrix back substitutions using L. The number of operations in a back substitution is proportional to the number of nonzeroes in L. It is assumed that the A originally has $O(n)$ nonzeroes. This is reasonable since the maximum adjacency list length is typically a small constant (e.g., 6). However, it is also assumed that A has the structure of an $n \times n$ Laplacian matrix, with a bandwidth of $O(\sqrt{n})$, and that no reordering scheme is used. Then the fill-in of L is $O(n^{1.5})$ nonzeroes, corresponding to fill-in between the bands. It is assumed that fill-in occurs at the rate $O(k^{1.5})$ per iteration as rows are added to L. Using the following relationship,

$$\sum_{k=1}^{n} k^{1.5} < \sqrt{n} \sum_{k=1}^{n} k = \frac{1}{2}\sqrt{n}(n^2 + n),$$

we bound the total computational effort as $O(n^{2.5})$. Thus, the complete CVFE algorithm requires $O(n^{2.5})$ operations. A CVFE algorithm which assembles and factors the stiffness matrix at each iteration would theoretically require $O(n^2)$ operations per time step and $O(n^3)$ in total. The introduction of reordering algorithms could further reduce the complexity of both the CVFE algorithm described in this paper as well as conventional approaches.

3 Numerical Results

This section summarizes implementation and performance details of the filling algorithm. The details include type of architecture and source language, method of validation, comparison with related codes, and timing results for several test problems.

The filling algorithm has been implemented in Fortran 77 for the Silicon Graphics (SGI) workstation architecture and given the code name ISOFIL. ISOFIL was developed under a systems integration plan based on the SGI Explorer program and makes use of extensions to Fortran 77, including the Fortran POINTER data type and the `malloc` procedure call. ISOFIL also includes routines from two public domain software libraries, BLAS (level 1) [13] and SPARSPAK [12]. All performance results reported in this section are based on the SGI model 4D-35 workstation.

Numerical validation of ISOFIL so far has included evaluation of mass balance. In one validation exercise, a 10 cm radius, center-gated, disk mold was discretized with 800 triangles and injected at the constant flow rate of one cc/second assuming a void fraction of 70%, anisotropic permeability ($k_{11} = 1, k_{22} = .3$), and two fiber orientations. The simulated filling time was 213.8 seconds compared to the expected (analytical) 219.6 seconds, or about 3% relative error. The same model was evaluated using a constant pressure injection of one kg/cm^2. The mass influx was approximated by the flux across the control volume boundaries surrounding the injector. The resulting approximation was 214.1 cc to fill the mold, versus the expected (analytical) 219.6 cc. The CPU time was 5 seconds, including input and output.

A qualitative evaluation of the simulated flow fronts indicates that flow front shape is determined by element shape. This result can also be inferred from inspection of Figure (2). Highly elongated elements lead to elongated control volumes. Elongated volumes may fill before neighboring volumes begin to fill, in a locally non-physical manner. This effect demonstrates the need for a relatively uniform mesh with a good aspect ratio for the triangles.

The performance of ISOFIL was compared to a predecessor code, LIMS, from the University of Delaware [7]. ISOFIL uses the same CVFE approach as LIMS to model pressure and fluid velocity. There are minor differences in how control volume flux is calculated. The primary differences are that LIMS assembles and factors the pressure stiffness matrix and the flux stiffness matrix at each time step of the CVFE algorithm. As a result, the performance differential between the two codes increases with n, the number of nodes. For a 2213-node, 4443-triangle model of the Ford Aerostar Crossmember (see below), the CPU times were 230 seconds for ISOFIL and 22389 seconds for LIMS 2.2, a speedup factor of approximately 100.

A set of four test problems are presented in Figure 4, including a plaque, disk, auto crossmember, and aircraft keel prototype. The plaque and disk models are simple two-dimensional geometries, while the crossmember and prototype keel box are thin-shell, three-dimensional structures. These models have been simulated under various conditions, including different choices of injector/vent locations, material type (permeability and fiber orientation). The performance results are presented in Table 1. The larger test problems require several minutes of CPU time. The time to read the input deck (element mesh and material properties) is included in the results, however it is not a major fraction of the total time. The error in mass balance ranges up to 3% for the test problems and is calculated as described above for the constant pressure injection case.

Problem	Nodes	Elements	Sec.	Mass Bal.	Speedup
disk	442	800	5	2.60%	23
plaque	925	1728	22	0.02%	33
xmbr	2213	4443	230	0.45%	97
proto	2066	4116	188	0.39%	na

Table 1: **ISOFIL Performance on Four Models.** Results were obtained on an SGI 4D-35, CPU time includes I/O. Speedup is the ratio of LIMS 2.2 CPU time to ISOFIL CPU time.

4 Conclusions

The isothermal RTM filling algorithm has a computational complexity of $O(n^{2.5})$, compared to more general RTM algorithms which also employ the CVFE formulation but require $O(n^3)$ computation. This relative advantage results in a 100-fold performance improvement over a similar code for a 2213-node model, while obtaining the same solution accuracy.

The speed of the filling simulation is critical to applying simulation results to actual mold design. A fast and flexible simulation tool allows engineers to include modeling in the design process. The ISOFIL code is currently being used for interactive filling simulation of structural aircraft components in connection with Army procurement projects. The complete software system permits interactive graphical manipulation of mesh and material properties, as well as the location and specification of injection pressure/displacement time profiles. The details of this complete system will be published in a future paper.

Interactive simulation (and real-time control) is now feasible for small ($n \approx 1,000$) problems on high performance workstations and supercomputers. However the computational requirement of $O(n^{2.5})$ operations still prohibits interactive simulation of refined three-dimensional models involving $n > 10^5$ nodes (massively parallel supercomputers are capable of about 10^{10} floating point operations per second). We are interested in simulations which require a matter of seconds or, at most a few minutes, of real time. This motivates the need for further investigation of filling algorithms which depart from the conventional CVFE strategy.

5 Acknowledgments

We acknowledge the advice and assistance of Mathematician *David J. Eyre*, at the Army High Performance Computing Research Center, University of Minnesota. R. Maier acknowledges the support of Army Research Office contract number DAAH04-93-G-0207.

References

[1] WALSH, SHAWN M., UNITED STATES OF AMERICA, *IN-SITU SENSOR METHOD AND DEVICE*, patent number 5,210,499, issued May 11, 1993, U.S. Patent Office.

[2] K. MEISSNER AND P. SINCEBAUGH, Preventing mechanical failures in resin transfer molding using embedded sensors and neural networks, preprint, U.S. Army Research Laboratory, Materials Directorate, Watertown MA 02172-0001.

[3] AVNER FRIEDMAN, *Partial Differential Equations of Parabolic Type*, Prentice-Hall, Englewood Cliffs, 1964.

[4] C. L. TUCKER AND R. B. DESSENBERGER, "Governing Equations for Flow through Stationary Fiber Beds," in *Flow and Rheology in Polymer Composites Manufacturing*, ed. S.G. Advani, Elsevier, 1994.

[5] H. P. WANG AND H. S. LEE, "Numerical Techniques for Free and Moving Boundary Problems," in *Fundamentals of Computer Modeling for Polymer Processing*, ed. C.L. Tucker III, Hanser, Munich, 1989.

[6] S. Advani and M. Bruschke A Finite Element/Control Volume Approach to Mold Filling in Anisotropic Porous Media, Polymer Composites, Vol. 11, No. 6, December 1990.

[7] S. Advani and M. Bruschke *Liquid Injection Molding Simulation User's Manual Version 2.0* Center for Composite Materials, University of Delaware, Newark, DE, 19716, December 1992.

[8] S. Advani, M. V. Bruschke, and R. S. Parnas, "Resin Transfer Molding," in *Flow and Rheology in Polymer Composites Manufacturing*, ed. S.G. Advani, Elsevier, 1994.

[9] W.B. Young, K. Han, L.H. Fong, L. J. Lee, and M.J. Liou, Flow simulation in molds with preplaced fiber mats,*Polymer Composites,* vol. 12, p.391, 1991.

[10] W.B. Young, L. J. Lee, and M.J. Liou, Modification of Control Volume Finite Elementr Method in Mold Filling Simulation, Report No. ERC/NSM-P-91, NSF Engineering Research Center for Net Shape Manufacturing, The Ohio State University, Columbus, 43210, Jan. 1991.

[11] Alan George and Joseph Liu, *Computer Solution of Large Sparse Positive Definite Systems,* Prentice-Hall, Englewood Cliffs, 1981.

[12] SPARSPAK,*NETLIB@ornl.gov.*

[13] Basic Linear Algebra Subroutines,*NETLIB@ornl.gov.*

[14] Thomas J. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Englewood Cliffs, 1987.

[15] J. Crank, *Free and Moving Boundary Problems,* Oxford University Press, 1984.

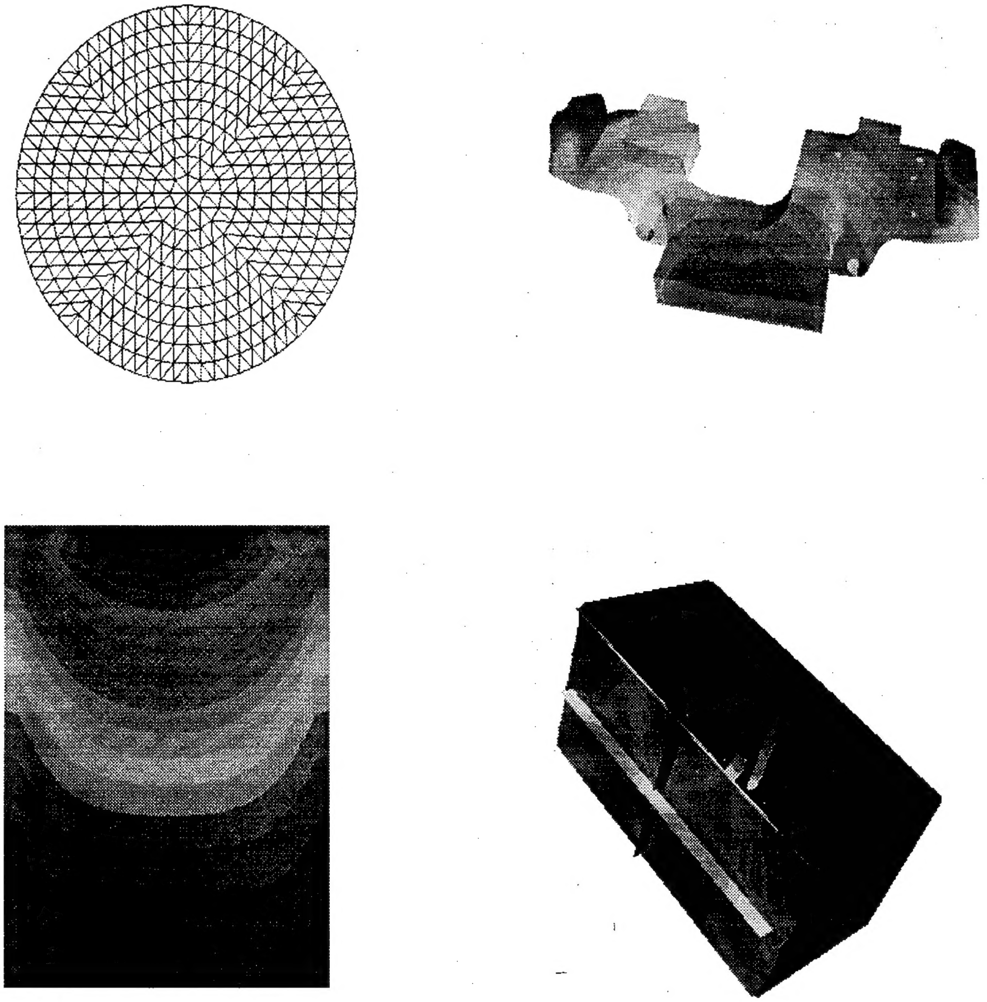

Figure 4: **Four Test Problems with Flow Front Histories.** Clockwise from upper left: disk, crossmember, keel prototype, plaque.